CSC 373: Algorithm Design and Analysis
Lecture 17

Allan Borodin

March 4, 2013

Some materials are from Keven Wayne's slides and MIT Open Courseware spring 2011 course at http://tinyurl.com/bjde5o5 .

Announcements and Outline

Announcements

- Lecture this Friday.
- Assignment 2 is due this Friday, March 1.
- Term test 2 on Monday, March 4.
- You must keep all graded work until the term is over just in case there is some inconsistency in the grades recorded and what you have.

Today's outline

- Review 3SAT $\leq_p$ SubsetSum transformation and consequences.
- 3SAT $\leq_p$ 3-COLOR
- Cook's proof that SAT is NP complete.
- **Note:** I am going to present the Turing machine model on the board and may show a simulation from the web. One such site is `http://morphett.info/turing/turing.html`

3SAT reduces to Subset Sum

Claim

3SAT $\leq_p$ Subset Sum

- Given an instance F of 3SAT, we construct an instance of Subset Sum that has solution iff F is satisfiable.

$$C_1 = \bar{x} \vee y \vee z$$
$$C_2 = x \vee \bar{y} \vee z$$
$$C_3 = \bar{x} \vee \bar{y} \vee \bar{z}$$

	x	y	z	C_1	C_2	C_3
x	1	0	0	0	1	0
¬ x	1	0	0	1	0	1
y	0	1	0	1	0	0
¬ y	0	1	0	0	1	1
z	0	0	1	1	1	0
¬ z	0	0	1	0	0	1
dummies to get clause columns to sum to 4	0	0	0	1	0	0
	0	0	0	2	0	0
	0	0	0	0	1	0
	0	0	0	0	2	0
	0	0	0	0	0	1
	0	0	0	0	0	2
W	1	1	1	4	4	4

Some consequences of SubsetSum completeness

- Recall hint for question 4b of assignment.
- SubsetSum $\leq_p$ Knapsack where

  $$\text{Knapsack} = \left\{(\langle s_1, v_1\rangle, \ldots, \langle s_n, v_n\rangle; S, V) \mid \exists S : \sum_{i\in S} s_i \leq S, \sum_{i\in S} v_i \geq V\right\}$$

- SubsetSum $\leq_p$ Half-SubsetSum where

  $$\text{Half-SubsetSum} = \{a_1, \ldots, a_n \mid \exists S, \sum_{i\in S} a_i = \tfrac{1}{2}\sum_{i=1}^{n} a_i\}.$$

- The NP completeness of Half-SubsetSum implies the completeness of a decision problem version of the makespan problem.

Reviewing how to show some L is NP complete.

- We must show $L \in$ NP. To do so, we provide a polynomial time verification predicate $R(x, y)$ and polynomial length certificate y for every $x \in L$; that is,

$$L = \{x \mid \exists y, R(x, y) \text{ and } |y| \leq q(|x|)\}.$$

- We must show that L is NP hard (say with respect to polynomial time transformations); that is, for some known NP complete L', there is a polynomial time transducer function h such that $x \in L'$ iff $h(x) \in L$. This then establishes that $L' \leq_p L$.

- **Warning:** The reduction/transformation $L' \leq_p L$ must be in the correct direction and h must be defined for every input x; that is, one must also show that if $x \notin L'$ then $h(x) \notin L$ as well as showing that if $x \in L'$ then $h(x) \in L$.

Some transformations are easy, some not

- Tranformations are (as we have been arguing) algorithms computing a function and hence like any algorithmic problem, sometimes there are easy solutions and sometimes not.
- In showing NP-completeness it certainly helps to choose the right known NP-complete problem to use for the transformation.
- In the Karp tree, there are some transformations that are particularly easy such as :
  - IndependentSet $\leq_p$ VertexCover
  - VertexCover $\leq_p$ SetCover
- A transforrmation of moderate difficulty is 3SAT $\leq_p$ 3-COLOR
- I am using Kevin Wayne's slides to illustrate the transformation. See slides for "Poly-time reductions" in

  http://www.cs.princeton.edu/courses/archive/spring05/cos423/lectures.php

3CNF $\leq_p$ 3-COLOR: Outline of Transformation

Claim. 3-SAT $\leq_P$ 3-COLOR.

Pf. Given 3-SAT instance Φ, we construct an instance of 3-COLOR that is 3-colorable iff Φ is satisfiable.

Construction.

i. For each literal, create a node.
ii. Create 3 new nodes T, F, B; connect them in a triangle, and connect each literal to B.
iii. Connect each literal to its negation.
iv. For each clause, add gadget of 6 nodes and 13 edges.

↑
to be described next

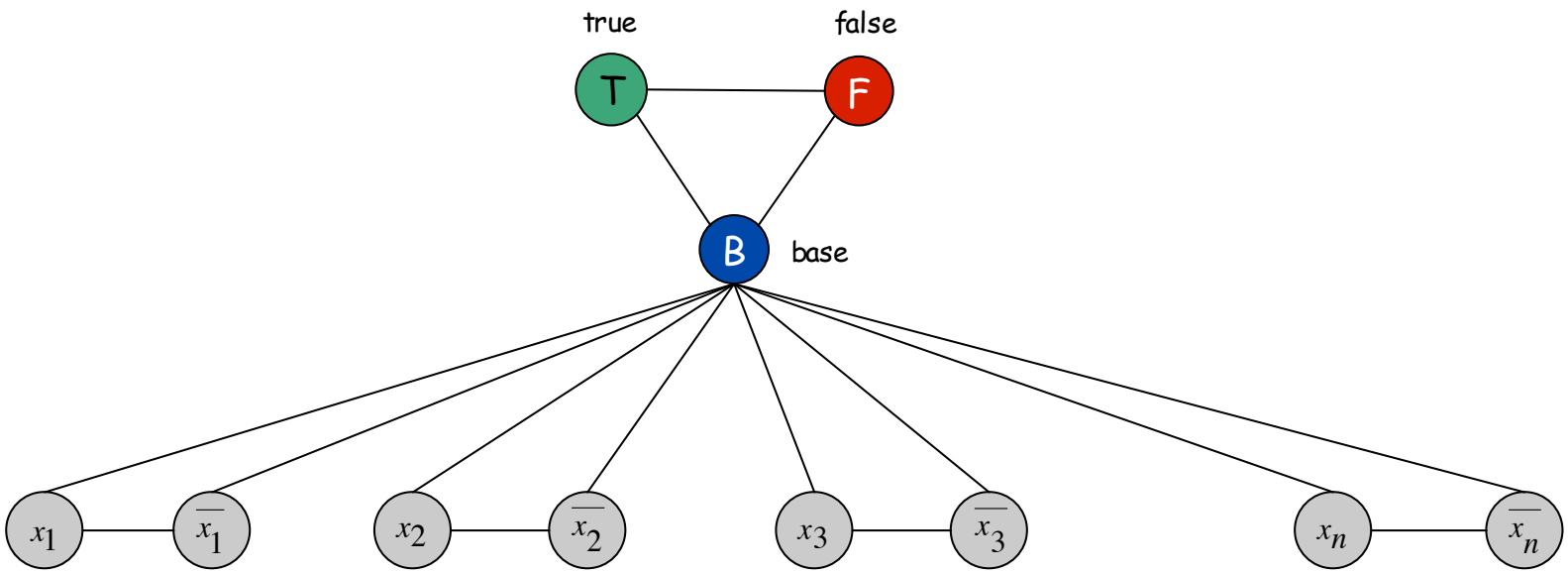
- If ϕ is a 3CNF formula in n variables and m clauses, then $h(\phi) = G_\phi$ will have $2n + 6m + 3$ nodes and $3n + 13m + 3$ edges.

3CNF $\leq_p$ 3-COLOR: Consistent literals

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.

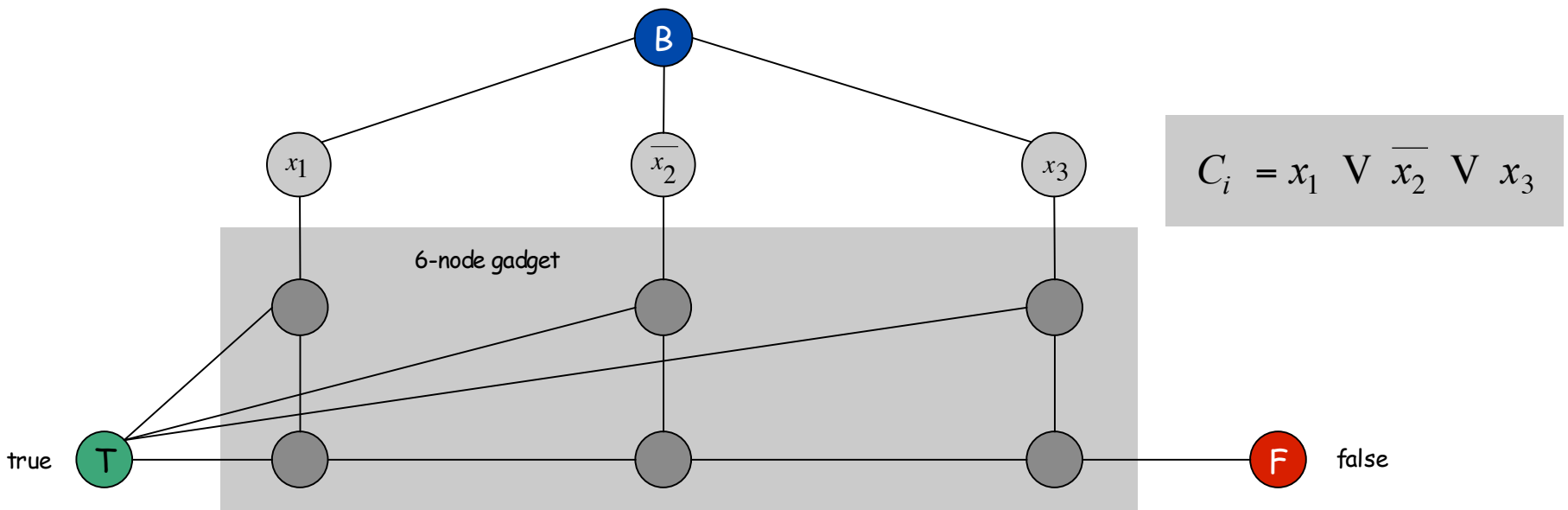


3CNF $\leq_p$ 3-COLOR: The clause gadget

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

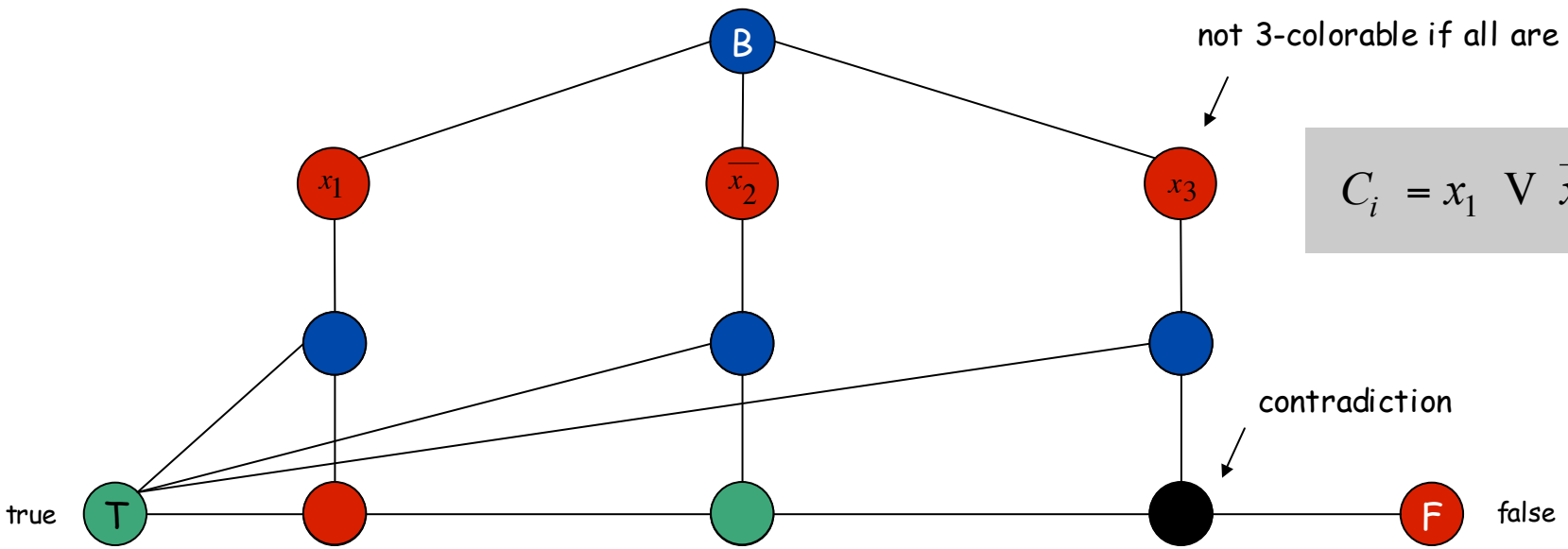


G_ϕ is 3-colourable $\Rightarrow$ ϕ satisfiable

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Rightarrow$ Suppose graph is 3-colorable.

- Consider assignment that sets all T literals to true.
- (ii) ensures each literal is T or F.
- (iii) ensures a literal and its negation are opposites.
- (iv) ensures at least one literal in each clause is T.

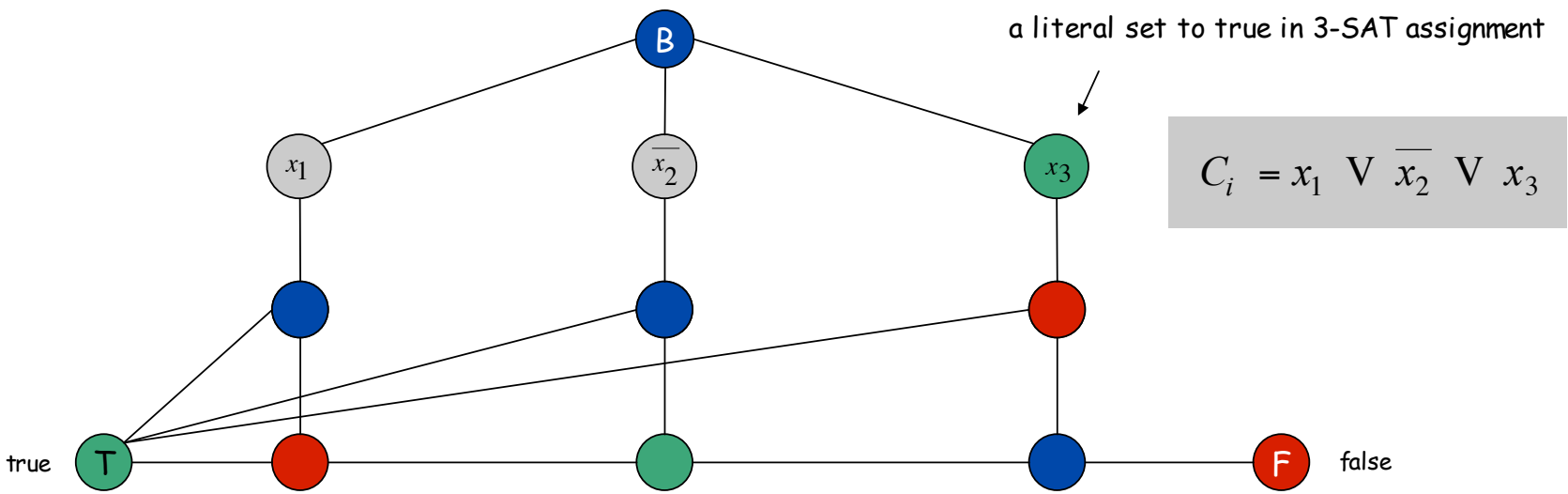


ϕ satisfiable $\Rightarrow$ G_ϕ is 3-colourable

Claim. Graph is 3-colorable iff Φ is satisfiable.

Pf. $\Leftarrow$ Suppose 3-SAT formula Φ is satisfiable.

- Color all true literals T.
- Color node below green node F, and node below that B.
- Color remaining middle row nodes B.
- Color remaining bottom nodes T or F as forced. ▪



Brief introduction to Turing machines

- We are using the classical one tape TM. This is the simplest variant to formalize which will enable the proof for the NP completeness of SAT. In the proof, we are assuming (without loss of generality) that all time bounds $T(n)$ are computable in polynomial time.
- Claim Any reasonable (classical) computing model algorithm running in time $T(n)$, can be simulated by a TM in time $T(n)^k$ for some k. Hence we can use the TM model in the definition of P and NP.
- Since we are only considering decision problems we will view TMs that are defined for decision problems and hence do not need an output other than a reject and accept state.
- Following the notation in the MIT lecture notes, formally, a specific TM is a tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$
- We briefly explain (using the board) the model and notation. Note that Q, Σ, Γ are all finite sets.